Re: PIC24F experience/hints

Source: http://coding.derkeiler.com/Archive/General/comp.arch.embedded/2007−11/msg02233.html

- *From*: Leon <leon355@xxxxxxxxxxxxxx>
- *Date*: Fri, 30 Nov 2007 11:20:18 −0800 (PST)

On 30 Nov, 06:00, jhal...@xxxxxxxxxxxx (Joseph H Allen) wrote:

> I've been playing with a PIC24FJGA002: a 16−bit, 16 MIPS PIC in a 28−pin DIP with 8KB RAM.
>
> Here are some quick−start hints:
>
> Programming: I bought a cheap ICD2 clone from ebay, one fromwww.mdfly.com They don't advertise that it works with this chip, but it does for both programming and debugging. However, you need to make your own programming socket (or do in−circuit). Unlike older PICs:
>
> You need a 10 uF capacitor between VCAP and ground. The PIC needs this to generate its internal 2.5V Vcore supply.
>
> You need to ground DISVREG
>
> Connect VPP to MCLR (MCLR is no longer labeled VPP− ICD2 will not put 13V on MCLR, just pull it to either 3.3V and GND).
>
> Set the MDFLY ICD2 clone to supply 3.3V (this is a jumper)− or set it to Tself, and use your target board to power the PIC.
>
> Connect PGC and PGD to any one of the three sets of PGCx and PGDx pins. It does not matter which of the three you pick− I used PGC1 and PGD1.
>
> Reset: Unlike some 8−bit PICs, you can not disable MCLR. It needs a pull−up for the PIC to come out of reset.
>
> The C30 C compiler works fairly well. Ints are 16−bits, like in MS−DOS. Here is a program which blinks a LED on RA0:

```
#include "pic24fxxxx.h"

void e()
{
}
```

```
void dly()
{
int x;
for (x = 0; x != 10; ++x)
e();
}

int main()
{
int x;
PORTA = 1;
TRISA = 0;
for (;;)
{
PORTA = 1;
for (x = 0; x != 10000; ++x) dly();
PORTA = 0;
for (x = 0; x != 10000; ++x) dly();
}
}
```

In MPLAB, create a project, select the device and select "FRC PLL" (built−in 8 MHz RC clock multiplied by 4 by PLL) config option. Build it, and burn it into the chip or debug it. Remember to select "Debug" or "Release" on the tool bar. It needs to be "Debug" for debugging (adds −g to GCC).

For programming, remember to hit to "release from reset" button after you program the part or nothing will happen :−)

For debugging (I mean emulating), a part of the programming tool bar and the debug tool bar appear. Hit the program button first, then hit the reset device button, and then debug: hit run, or right−click and hit goto cursor or whatever.

Single−stepping is incredibly slow− I wonder if it has to reprogram the flash after each step.

I'm in my 30−day eval period for the C30 compiler, after which it still works but prevents optimization. This sucks (especially since it's gcc), because even for the program above there is a 3X performance increase between no optimization and −O3.

I have AVR tools... I'll have to do a speed comparison.
−−
/* jhal...@xxxxxxxxxxxxx AB1GO */ /* Joseph H. Allen */

```
int a[1817];main(z,p,q,r){for(p=80;q+p−80;p−=2*a[p])for(z=9;z−−;)q=3&(r=time(0)
+r*57)/7,q=q?q−1?q−2?1−p%79?−1:0:p%79−77?1:0:p<1659?79:0:p>158?−79:0,q?!a[p−+q*2
]?a[p+=a[p+=q]=q]=q:0:0;for(;q++−1817;)printf(q%79?"%c":"%c\n"," #"[!a[q−1]]);}
```

Single−stepping with the ICD 2 is slow, especially if you are

displaying registers. It's nothing to do with programming the flash as
the instructions are moved into RAM when they are executed.

Leon
.